第二次作业

第一章

一. 名词解释

1. 操作系统

答: 操作系统是一种系统软件，其主要功能是管理计算机系统的硬件与软件资源，为应用程序的运行提供支持与服务。操作系统处于用户与计算机硬件之间，作为资源管理者和控制程序，负责协调处理器、内存、输入输出设备、文件系统等资源的分配与使用，确保系统的有效性、并发性安全性与可扩展性.

2. 预输入

答: 预输入是指操作系统在程序实际请求数据之前，基于访问局部性原理，提前将可能使用的页面或数据从外存加到内存，以减少缺页中断或I/O等待，提高系统效率.

3. 分布式操作系统

答: 分布式操作系统是运行在分布式系统之上的系统软件，负责对分布环境中的多个独立计算节点进行统一管理和协调控制，为用户提供一个逻辑上单一、整体、一致的操作环境

4. 实时操作系统.

答: 系统能及时响应外部事件的请求，在规定时间内完成对该事件的处理，并控制所有实时任务协调一致地运行.

5. 互斥共享

答: 互斥共享是指多个进程对同一共享资源访问时，在任一时刻仅允许一个进程访问，以防止并发操作导致资源冲突或数据不一致。

三、填空题:

1. 硬件，软件 2. 裸机 3. 硬件资源，软件资源 4. 顺序性 5. 共享性，异步性
6. 进程控制，调度 7. 程序接口 8. 单用户单任务，单用户多任务，多用户多任务

四. 判断题

1. × 2. √ 3. √ 4. √ 5. × 6. √ 7. √ 8. √ 9. × 10. √

第二章

一. 名词解释

1. 进程上下文

答: 进程上下文是指操作系统为正确管理和切换进程而保存的与进程相关的所有信息, 主要包括用户上下文, 内核上下文以及硬件上下文.

2. 直接制约

答: 直接制约是指进程间通过共享数据或资源而建立的直接同步与互斥关系, 即一个进程的执行必须等待另一个特定进程的某种状态变化.

3、临界区

答: 临界区是指程序中访问共享资源的一段代码。在同一时刻, 只允许一个进程进入临界区执行, 以防止因并发访问造成数据不一致或冲突.

4. 进程同步

答: 进程同步是指多个进程在并发执行过程中, 为了协调对共享资源的有序访问或实现特定的执行顺序, 而在执行过程中相互等待, 协同运行的一种制约和制.

5. 内核线程

答: 内核线程是由操作系统内核管理和调度的线程, 其创建、终止与调度完全由内核负责, 可以在多个处理器之间并发执行, 并支持内核级的线程切换与资源分配.

三. 填空题

1. 动态性, 独立性, 异步性　2. 运行态, 执行态　3. 进程调度信息, 进程控制信息
4. 系统空间　5. 动态, 静态　6. 间接制约　7. 空闲让进, 忙则等待, 有限等待, 让权等待
8. 管道, 消息队列, 内存, 共享　9. 执行状态, 就绪状态, 阻塞状态　10. 邮箱

四. 判断题

1. √　2. √　3. √　4. √　5. ×　6. ×　7. √　8. √　9. ×　10. √

五. 简答题

1. ①"忙等"是指一个进程在等待某一条件满足时，不让出CPU，而是不断占用处理器反复检查条件是否满足

其缺点是：浪费CPU资源，容易造成系统效率下降，特别是在资源紧张时，不利于多进程并发调度.

②"饥饿"是指一个进程长时间得不到所需资源，导致迟迟不能执行甚至永远等待下去

其缺点是：降低系统公平性，某些关键进程可能永远无法执行，造成逻辑错误或系统死锁风险

2. ① S1：无前驱，是起始进程；S2：前驱为S1；S3：前驱为S1；S4：前驱为S2；

S5：前驱为S2和S3；S6：前驱为S3；S7：前驱为S4、S5、S6.

② 信号量命名为Semi表示Si的同步信号；~~Si~~ Si_j表示 Si→Sj的同步信号

```
semaphore S1_2=0, S1_3=0;
semaphore S2_4=0, S2_5=0;
semaphore S3_5=0, S3_6=0;
semaphore S4_7=0, S5_7=0, S6_7=0;
process S1(){
    do_S1();
    signal(S1_2);
    signal(S1_3);
}
process S2(){
    wait(S1_2);
    do_S2();
    signal(S2_4);
    signal(S2_5);
}
process S3(){
    wait(S1_3);
    do_S3();
    signal(S3_5);
    signal(S3_6);
}
process S4(){
    wait(S2_4);
    do_S4();
    signal(S4_7);
}
process S5(){
    wait(S3_6);
    do_S6();
    signal(S6_7);
}
process S7(){
    wait(S4_7);
    wait(S5_7);
    wait(S6_7);
    do_S7();
}
```

3. 信号量设计

mutex：互斥访问盘子的信号量，初值=1

empty：空位数，初值=5；

apple：盘中苹果数，初值=0；

orange：盘中桔子数，初值=0；

```
Father() {
    while(true){
        produce_fruit();
        p(empty);
        p(mutex);
        if (fruit==APPLE)
            put_apple(); signal(apple);
        else
            put_orange(); signal(orange);
        V(mutex);
    }
}
Son() {
   while (true) {
        p(orange);
        p(mutex);
        take_orange();
        V(mutex);
        v(empty)
        eat();
    }
}
```

```
Daughter( ) {
    while(true) {
        p(apple);
        p(mutex);
        take_apple();
        V(mutex);
        V(empty);
        eat();
    }
}
```

4. 信号量设计：

S1：控制PA执行(初值1)；

S2：控制PB执行(初值0)；

S3：控制PC执行(初值0)；

Semaphor S1=1, S2=0, S3=0;

```
PA() {
    while(true){
            P(S1);
            write_to_buffer1();
            V(S2);
        }
    }
PB() {
    while(true){
        P(S2);
        copy_buffer1_to_buffer2();
        V(S3);
    }
}
PC() {
    while(true){
        P(S3);
        print_buffer2();
        V(S1);
    }
}
```

5、信号量设计：

mutex：对候客区操作的互斥，初值=1；

customers：等待服务的顾客数（初值=0）；

barbers：理发师是否空闲，初值=0；

waiting：候客椅数量计数器（设为N）.

```
int waiting=0;
int N=5;
Semaphor mutex=1;
Semaphor customers=0;
Semaphor barber=0;
Barber(){
    while(true){
        P(customers);
        P(mutex);
        waiting--;
        V(barber);
        V(mutex);
        cut_hair();
    }
}
Customer(){
    P(mutex);
    if(waiting<N){
        waiting++;
        V(customers);
        V(mutex);
        P(barbers);
        get_haircut();
    }else{
        V(mutex);
        leave;
```

第三章

一、名词解释

1. 作业

答：作业是用户提交给计算机系统要求完成的一项或多项任务的集合，通常包括程序、数据及作业说明书，是操作系统资源调度与管理的基本单位

2. 处理机调度

答：处理机调度是操作系统在多个就绪进程之间选择一个进程，并将CPU资源分配给该进程的执行的过程，是实现进程并发执行和系统资源合理利用的核心机制之一。

3. 周转时间

答：从作业提交给系统开始，到作业完成为止的这段时间间隔。

4. 死锁

答：指多个进程在运行过程中因争夺资源而造成的一种僵局，当进程处于这种僵持状态时，若无外力作用，这些进程都将永远不能再向前推进。

5. 临时性资源

答：临时性资源是指使用后可以立即释放，并可被其他进程重复利用的系统资源，进程使用这些资源具有占用时间短，使用时简单，不需长时间保持等特点。

三、填空题

1. 脱机输入，联机输入，直接输入，远程输入　2. 后备，就绪，完成　3. 高级调度，低级调度，内存调度，交换调度

4. 静态，动态　5. 硬实时，软实时

6. 预防死锁，避免死锁，检测死锁，解除死锁。

四、判断题

1. × 2. × 3. √ 4. × 5. √ 6. × 7. × 8. √ 9. √ 10. √

五、问答题

1. FCFS:

进程	A	B	C	D	E	平均
到达	0	2	4	6	8	
服务	3	6	4	5	2	
完成	3	9	13	18	20	
周转	3	7	9	12	12	8.6
带权	1.00	1.17	2.25	2.40	6.00	2.56

各进程的等待时间为: A:0, B:1, C:6, D:8, E:11

2. SPF(非抢占)

进程	A	B	C	D	E	平均
到达	0	2	4	6	8	
服务	3	6	4	5	2	
完成	3	9	15	20	11	
周转	3	7	11	14	3	7.6
带权	1.00	1.17	2.75	2.80	1.50	1.84

各进程等待时间为:
A:0, B:1, C:8, D:10;
E:2

3. SPF(抢占):

进程	A	B	C	D	E	平均
到达	0	2	4	6	8	
服务	3	6	4	5	2	
完成	3	15	8	20	10	
周转	3	13	4	14	2	7.2
带权	1.00	2.16	1.00	2.80	1.00	1.59

各进程等待时间为: A:0, B:7, C:0, D:9, E:0

4. HRRN:

进程	A	B	C	D	E	平均
到达	0	2	4	6	8	
服务	3	6	4	5	2	
完成	3	9	13	20	15	
周转	3	7	9	14	7	8
带权	1.00	1.17	2.25	2.80	3.50	2.14
等待	0	1	5	9	5	4
带权	0	0.17	1.25	1.8	2.5	1.14

5. RR:

进程	A	B	C	D	E	平均
到达	0	2	4	6	8	
服务	3	6	4	5	2	
完成	4	18	17	20	15	
周转	4	16	13	14	7	10.8
带权	1.33	2.67	3.25	2.80	3.50	2.71
等待	1	10	9	9	5	6.8
带权	0.33	1.67	2.25	1.8	2.5	1.71

2. 设任务A、B、C在t=0时同时到达，任务A和B每次必须完成的时间分别为：

A1，A2，A3...和B1，B2，B3，...

t=0

A1须在20ms时完成，其本身运行时间是10ms，

A1的松弛度=(20-10-0)ms=10ms

B1的松弛度=(50-10-0)ms=40ms

C1的松弛度=(50-15-0)ms=35ms

所以可得到A1先执行，当A1执行完10ms后，只剩下了B1和C1，此时

t=10ms

B1的松弛度=(50-10-10)ms=30ms

C1的松弛度=(50-15-10)ms=25ms

所以可以得到C1先执行，当C1执行到了

t=25ms时

A2的松弛度=(40-10-25)ms=5ms

B1的松弛度=(50-10-25)ms=15ms

所以可得到A2先执行，当A2执行完10ms时

t=35ms，只剩下了B1，接着执行B1，当B1执行完10ms时，

t=45ms，只剩A3，执行A3，当A3执行完10ms时，

t=55ms，此时

B2的松弛度=(100-10-55)ms=35ms

C2的松弛度=(100-15-55)ms=30ms 所以C2执行15ms

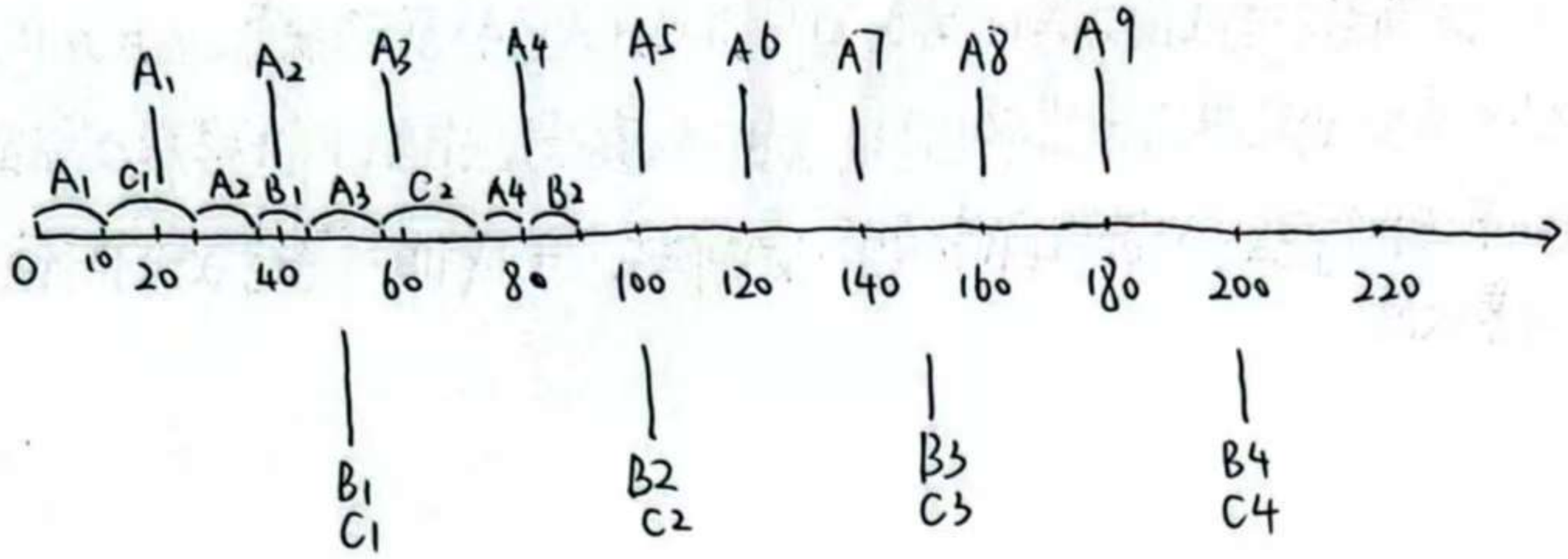
此时，t=70ms

A4的松弛度=(80-10-70)ms=0ms，则A4执行10ms

此时，t=80ms，只剩下了A5和B2

A5的松弛度=(100-10-80)ms=10ms

B2的松弛度=(100-10-80)ms=10ms ...



3. (1)

process	Allocation
P0	0032
P1	1000
P2	1354
P3	0332
P4	0014

	Work	Need	Allocation	Work+Allocation	Finish
P0	1622	0012	0032	1654	True
P3	1654	0652	0332	1986	True
P4	1986	0656	0014	19910	True
P1	19910	1750	1000	29910	True
P2	29910	2356	1354	3 12 14 14	True

利用安全性算法进行分析，找到了一个安全序列 [P0, P3, P4, P1, P2]，因此系统是安全的

(2) P2发出请求 Request(1,2,2,2)后，系统按银行家算法进行检查。

① Request(1,2,2,2) <= Need(2,3,5,6)

② Request(1,2,2,2) <= Available(1,6,2,2)

③ 系统先假定可为P2分配资源，并修改Available, Allocation, Need向量

Available=(0,4,0,0)　Allocation=(2,5,7,8)　Need=(1,1,3,4)

④ 进行安全性检查，此时对所有的进程，条件Need <= Available(0,4,0,0)都不成立，即Available不能满足任何进程的要求，则系统进入不安全状态。因此当进程P2提出请求 Request(1,2,2,2)后，系统不能把资源分配给它。

(3) 系统立即满足进程P2的请求(1,2,2,2)后，并没有马上进入死锁状态。因为此时上诉进程并没有申请新的资源，并因得不到资源而进入阻塞状态。只有当上诉进程提出新的请求导致所有没执行完的多个进程因得不到资源而阻塞并形成循环等待链的时候，系统才进入阻塞状态.